CMPT 275 (02-2) **ASSIGNMENT #1** **Revised: 02/05/15**

DUE DATE: At the START of class, Wednesday, June 5, 2002.

This first assignment involves groups of 4 students writing an approximately 20 page long Requirements Specification for the project described below. There is a rough outline of the table of contents of a requirements spec in Section 4.8 of the lecture notes. I will also put some examples of past Cmpt 275 student requirement specs on reserve in the library.

This assignment doesn't sound like a lot of work, but finding convenient meeting times, and extensively brainstorming, researching, analyzing, and agreeing on a spec is very time consuming (Don't Underestimate). Also, you have to do your portion of the document early enough that someone else can proofread it, and you have time to edit their suggested changes, and have time to give them to the project leader for integration into the final document.

Immediately read this whole assignment. Immediately after forming a team in class, agree on your first meeting time. Second, plan to meet a TA who will be role- playing a customer. Don't just come to the first meeting with the 'customer' and ask the customer "What are the requirements?". Instead, first sketch an initial Object Relationship Diagram (called a 'Class Diagram' in UML). This diagram should be transformed into 3rd Normal Form (3NF). Also, write a list of questions ahead of time, before meeting with the TA..

You all MUST read the Document Planning Guide in the appendix of Section 4 of the lecture notes. You should then immediately plan a group document-formatting standard of the specification among group members, all the sections will look the same. Put your group standard into an example MS-Word document or Word template file and distribute it to all team members. Finally, be prepared to merge and check your work with other group members 6 days before the due date to allow for proof reading and corrections. Each person's work should be reviewed by at least one other person. It is best if the proof-reader not correct the mistakes, but instead mark up the author's section of the document, then have the original author correct the mistakes. This way, the original author learns from his/her mistakes, and in addition, will not produce shoddy work and expect someone else in the team to clean it up. Don't forget to review the document not just for content, but for compliance with the team's document formatting standards. Four days prior to the deadline, the proof reader should return the reviewed and marked up sections to their authors. The author should correct their sections, and 2 days prior to the deadline submit them to the project leader/document coordinator for that assignment. The coordinator will integrate the sections into a final document. The final document should be given a final proof read by one or more team members (who have just RE-read this assignment), in enough time before the deadline for any needed final corrections. Finally, all team members should sign the cover page of the document, and the cover page of the release.

The required system will only be briefly described here. You will have to analyze the system by using both your knowledge (or hypothetical knowledge) of the application domain, and also seeing the TA/course instructor who will 'role play' a middle manager in the customer's company. You are to make reasonable assumptions, decisions, performance/capacity estimates, and then word the requirements document as if you had done extensive research, interviews, and reviews at the customer's premises. Rather than asking your 'customer' how the system should work (maybe he just bought the corporation and doesn't exactly know what he needs), you should instead propose several alternatives and ask for his opinion of the relative merits of each. You should also not ask "What do you think of our initial object relationship diagram?" Instead, ask detailed questions about the classes, relationships, cardinalities, and attributes. Also, ask or decide which subset of features is most likely to need prototyping to nail down the specs early. Also, ask about what long term future features may need to have an foundation engineered in now. The customer will be glad to give you his opinion on all such matters, and on whether you are biting off more than you can chew in a single semester. Also, feel free to 'make up' some specifications for the purposes of this Cmpt 275 exercise (like whether it is anticipated

that the product be subsequently marketed to other similar customers. Or what kind of printer the customer has).

NOTE: If any group wants to propose an altogether different project from that described below, that will certainly be considered and you should see the course instructor immediately.

BRIEF SYSTEM DESCRIPTION:

This semester I want you to specify and design an airfreight load management system. Its purpose is to manage the items of airfreight (both standard containers, and odd sized pallets) that are assigned to airfreight flights. Airplane model types (i.e. model designators like 'Boeing 747') with specific remembered size and carrying capacity are associated with each particular flight. Flight designators are a specially formatted 10-character string specifying the flight number and date. For simplicity, the departure time, originating airport, and destination of each flight need not be stored. You can assume that the airline has only one destination for cargo loading at the user's point of origin.

The airline company's customers bring or truck to the airport cargo items that fall into two categories:

a) Airfreight containers are all a single standard size: 2 meters wide, 1.5 meters high, and 1.75 meters long. They are designed to be latched to the aircraft cargo floors, each taking up a 2 meter wide by 1.75 meter long footprint. All aircraft have 6 meter wide cargo floors, so containers can be loaded 3 footprints wide across the floor.
b) Pallets are about a 20 cm high wood platform. They come in various widths and lengths and are usually lifted with forklifts. Miscellaneous cargo is lashed to the pallet by the airline's customers. The pallet is loaded in the aircraft and latched in place on the cargo floor.

The load management system is designed to prevent the assignment of more cargo weight than the assigned aircraft can carry, and of more cargo footprint space that the assigned aircraft has in its cargo hold.

Here is some (not all) of the typical 'use cases' are:

1) A customer trucks items to the air cargo building of the airport. Using a menu-oriented system, the airline clerk types in the necessary physical characteristics of an item. Only weight and customer name are required for containers brought in by existing customers. The system assigns the item an airwaybill ID number.
2) Later, the clerk decides to assign cargo items to a flight. He inquires of the system how much capacity is remaining on a particular flight. The system determines this info and presents 3 results: the total number of footprints free, the total remaining weight which can be assigned before the aircraft would become overloaded, and the largest rectangular pallet that could be assigned to the remaining space.
3) The clerk assigns items to the next flight one at a time until the flight is full. Depending later on whether you assign items to flights or flights to items, you may need another user function that lists all unassigned cargo items so as to pick one to assign to a flight.
4) Before the flight, the allocated aircraft breaks down. Another aircraft model must be able to be allocated to the flight with just a single simple user operation. Any cargo that does not fit on the new model should be de-assigned from the flight. The prototype release of the product could perhaps just reject an operation that requires de-assignment.
5) For the flight, a printed report must be generated on that flight's cargo for use by the loaders and pilot.
6) On completion of the flight, the flight is deleted. This also deletes all items that were on that flight from the system.

To simplify for Cmpt 275, you needn't include use case operations to 'inquire' about objects other than a particular flight (i.e. not inquire about airplane capacity). Nor to modify objects (i.e. change the characteristics of an item or airplane); instead you may assume it ok to delete and add a different replacement. Also, you will

not have to deal with dates, number of days in each month, or leap years, or the system clock. But in real life you would. Feel free to make a number of other simplifying assumptions throughout the project. If you are unsure, check with the TA or Instructor.

List and briefly describe each user operation. In addition, for each of the use cases, list (but do not bother elaborating on) the exceptions that can occur. Not C++ or Java exceptions, but operational exceptions like out of disk space, aircraft model assigned to flight does not exist, or customer already exists. These last two are called referential integrity and key uniqueness exceptions. This information is needed for the author of the future user manual. You will loose a small amount of marks for each exception not mentioned. Later in Assignment #3 I will also ask that you list other exceptions that will be thrown by I/O functions like reading "xyz" into an integer.

There are up to 10 flights per day, and data is stored for up to a month in advance. To simplify the system for CMPT 275, you will not have to deal with dates, number of days in each month, or leap years. But in real life you would. Aircraft can be up to 100 footprints in length.

Note that for simplicity in Cmpt-275, assume the airline is small and only requires a single user system. Only one clerk will handle airfreight data entry, and delete flights once they have reached their destination. There is therefore not much use for a Use Case diagram which shows which kinds of users are allows to do which kind of user operations.
The airline does not want to store widths or lengths for regular containers as that would require extra typing by the busy clerk. Since a large proportion of the airfreight are standard size containers, this dimensional information would also take up a lot of unnecessary disk space when all containers are basically the same size. Note that a customer of the airline's often has many air freight items in the system, and you do not want to wastefully store the customer info for each item.

PROTOTYPE AND FUTURE RELEASES:
Your requirements spec should describe the full system. But it should also list, in a separate section, what could be left out of an initial prototype implementation used to validate only the core aspects of system operation. To reduce the workload in later Cmpt 275 assignments, you may later (when the instructor specifies) be allowed to leave out some features as described below. These would thus be good things to identify, in the requirements spec, as being possibly unnecessary in a prototype:

a) A flight inquiry operation may not be needed in the first prototype.
b) De-assigning an item from a flight may not be necessary. It's purpose is to deal with items that are not delivered to the airport on time to be loaded, or to shift cargo to another flight to make better use of the space for a big pallet.
c) The entire concept of aircraft characteristics, which are stored independent from flight info, could be left out (this might denormalize your 3NF back down to 2NF).

Also, in a different sub-section, you should mention long range plans for future features that might be required in the coming years. This will allow the designers to choose an architecture that will allow easy later enhancement in those areas. Though YOUR job as an analyst is to think of these feature lists yourself (and enquire if/when they might be needed), I will get you started by giving you several. In your assignment, describe in at least a full paragraph the individual characteristics of one of these future features that you feel will eventually have to be added in the next couple of years:

- generate a printed report containing all items that a particular customer has on a particular flight.
- generate a cost for each item based on a linear combination of total weight and footprint area.

In addition to a paragraph on one of the above, add at least one other description of a feature that you think up.

GENERAL ADVICE:

The project can become very complicated very fast, so it is important to keep the system simple! Ask the customer what his MINIMUM requirements are! DO NOT choose a fancy graphical user interface unless you have two excellent GUI programmers on your team and have permission of the instructor. DO NOT use pull down menus or another screen-position-oriented user interface, unless you get special permission of the instructor. This is a one semester project so simply scrolling out a list of menu choices in a DOS command window, and having the user type the number/letter of a menu item (followed by a <cr>) is adequate. Similarly, don't try now to decide whether files will be sorted, or data stored in trees. Those are design, not requirements issues.

You will have to form a group who want to work in the same manner, and work on the same computer and in the same language. The instructor will help with this in class one day by having students move to various corners of the classroom. It is best if you do not work with a group that is completely made up of homogeneous friends, as you likely all have the same interests, talents, and weaknesses! You need members with different interests and talent: conceptualists, rigorous detail specifiers, high level designers, low level systems programmers, and several members with good English! Think about how you would describe yourself before next class. Which of these categories do you fall into? When forming teams in class, immediately GET EACH OTHER'S NAME, PHONE NUMBER, ADDRESS, E-MAIL address, then agree on a first meeting time and place. There is a form you can fill out during the class that is attached to the back of this assignment. You can get one extra copy of the form from the instructor to return to the instructor (so he knows who is working together in each team).

You are required to decide which person in the group will lead the group for each assignment. The project leadership is to rotate to a different group member with each new CMPT 275 assignment. Choose group leaders appropriate for each assignment: specification (application concept and documentation skills), user manual (application view and writing skills), design (language import/export knowledge skills), implementation (programming skills), and testing (debugging skills). The leader for a particular assignment should not take on as much early work on the assignment. Instead, they contact everyone and try to set up the meeting times, and they need to spend more time coordinating and assembling the rest of the team's work near to the end of the 2 week assignment period! In addition, the leader shouldn't do all the checking him/herself, but coordinate the delegation of it to others. However, the project leader will likely do the final merge of the file segments. Also, the current leader is required to take brief notes/minutes at meetings, and afterwards immediately photocopy or e-mail the minutes out to the members (some of whom may not have been able to attend that meeting). The minutes should mention major decisions, assignments of tasks to individuals, and when the deliverables from each individual's task have been AGREED/promised to be returned to some particular other group member for review and/or integration with the other work being done by the team. This is essential to set! Every time in past semesters that groups got in trouble for one reason or another, this is one of the things they expressed that would have helped: WRITTEN task allocations and delivery dates! This also forms a record that a group in conflict can take to show the instructor.

A common problem with 3 person groups is that one person gets a wild idea, another doesn't want to go along, and the third is neutral (either not knowledgeable on that subject, or just meek in personality). Four person teams seem to work better since if someone feels like going in some strange design direction, there are usually 2 sane members willing to speak out and veto the idea. The leader is not a dictator. Teams should vote on difficult specification and design issues, or consult the TA/instructor. The leader is a coordinator/advisor/checker. If anyone (even if out-voted) thinks things are going serious wrong (either technically or in the team personal dynamics), see/email/call the instructor who will try to assess if the group is off track or otherwise needs some help.

Also realize you will be working with several different types of personalities. Some will be very competent, others weak. Try to monitor progress by requiring a partial deliverable from each member every 3-6 days

before the assignment is due. Then if it is not done, the rest of the group will at least be alerted and can take management action (set a new deadline, and either apply a little pressure and/or help out!). Some students will point fingers and blame everyone else for their shortfalls in delivery. In some cases, this will be justified while in others this is just an excuse. Determining which is the case is very hard sometimes. Some people accept criticism ("suggestions") poorly, others with meek personalities unfairly take blame and extra work. Some students will be callous and others very sensitive. If when discussing something, someone goes red in the face, it may be they are about to get mad at you, or that they feel so bad that they are almost ready to cry. In either case, try not to get upset yourself, as then you double the problem. Finally, try to recognize those with good organizational ability. They may not be the best designers and programmers, but may be the best coordinators/planners/leaders. Though during the first assignment you will likely not know all the facets of your team members, be on the look-out for them. Be sensitive to them. Use your observations later to wisely help the team function. Don't take unfair advantage of them, and if you have a quiet personality, try not to let yourself be taken advantage of. These personality issues are part of everyday team management in industry. Most groups say they learned VERY interesting lessons about people (rather than just about software engineering) during their 275 project. Again, feel free to consult the instructor about personnel problems, either in confidence, or as a group. Also, if you want to consult the instructor, do it EARLY! If you come to him near the end of the course, or near the end of an assignment, it leaves him little time or flexibility to step in and manage the problem.

Try to arrange to meet REGULARLY at least twice a week, say for lunch every Monday, Wednesday, and Friday. Use E-mail a lot. See ACS handout M-12 on how to set up an e-mail team list, then mail the listname to the instructor and TA. Even if you work mostly at home, you should still dial into SFU e-mail to check your mail from each other, and for hints from the instructor. We all use e-mail a lot in 275! Students sometimes e-mail the TA or instructor and suggest times they would like to come and see him. If he has the address of the group or of every member, he can e-mail back everybody at once to confirm an appointment time (it also helps to carbon your whole team on the original mail so the TA can simply 'Reply To All' when confirming an appointment. Sometimes document style questions, and design/programming language questions can be asked and answered quickly via e-mail. Feel free to see the TA or instructor in his/her office hours, but also realize e-mail may save you time and travel. Also, the TAs may be willing to meet with you outside of office hours for the critical first assignment.

Read and heed the Document Planning Guide handed out by the instructor. Rather than writing your sections of the Requirements Spec, then trying to glue them together and put all sections into a common format, save yourself some work and agree on a 'team' format ahead of time. Also, follow fairly closely the requirements spec table of contents in the lecture notes (though drop sections that are inappropriate). Give the customer's company a fake name, your project a fancy system name, and your team a company name! Do not put all your diagrams at the end of the document. They should be interspersed with the relevant text (though they may be on separate pages inserted in). Have one or two team members glance at some of the example old 275 projects that will be put on reserve in the library, but follow the guidelines specified here as the ones on reserve are from different semesters and some even from different instructors.

Finally, start thinking about document and source code version control. Much to my glee, because it is such a vivid demonstration, groups have got messed up by two members modifying the same file (i.e. modifying other member's files) at night, and then exchanging files or disks the next day to update the group with the latest document section or code file. The result is that only some of the changes get into the final draft. Read the Configuration Management Tutorial in the appendix of Section 4 of the course notes. Finally, make sure at the end of each assignment, that each of you gets a complete copy of the documentation and of the files. This allows each of you to work on the next assignment before the previous one is marked and handed back. It also prevents the project from falling apart if someone on your team drops out or is sick/injured.

You don't need to have decided on the platform (CPU/OS) the product will run on to write the requirements spec (unless the customer has a particular computer already in use). Also, be careful not to try too many new things at once! Several semesters ago, one group tried a new language, a new development environment (debugger, linker, editor), and new word processor all at once in their 275 project. Both in Cmp-275 and in industry, this is a recipe for a late project! This particular group was penalized for handed in every assignment late!

If you propose to do your project on a home PC, you are entirely responsible for failure of your hardware, software, and debugging any strange things that you cannot demo up at SFU. If you have a disk crash the night before an assignment is due, I cannot drive over to your house to verify this and thus give you an extension. Make sure you keep backups in case your disk crashes! Also, if any of the university computers fail for less than 2 hours just prior to an assignment, I will not give an extension. You shouldn't be leaving things this late. Pretend you are writing the spec as part of a contract bid to IBM in Toronto, and bids have to make it to the courier's airplane by the assignment due time, otherwise the spec will miss the flight and the bid will be disqualified. As in a real company, things in Cmpt 275 should not be left until the last minute because the system or printer could go down and you would miss the courier on a $1 Million dollar contract bid.

GETTING STARTED ON THE REQUIREMENTS SPEC:

Draw the level 0 DFD to nail down the input and output sources.
Analyze the nouns and verbs in the system description above, and draw an Object Relationship Diagram (ORD). Put in 3rd normal form to find out how many files you will have, to show the key attributes, and to specify why you chose the given cardinalities (you MUST in a sentence each justify why the opposite optionality and opposite multiplicity are not correct). Do this for EACH END of each relationship! You probably shouldn't add object member functions yet (wait until Assignment 3). However, you should add relationship designator numbers (e.g. "R13") and foreign keys. Then add foreign key designators (e.g. "(R13)" to the foreign key attributes).

In previous semesters, I have asked students to decompose the context diagram into level 1 logical DFDs. This practice is falling out of favor in industry so I will eliminate this requirement.

You will likely find in 275 that some main object seems to transition amongst various states. Diagram its life cycle as a finite state machine. Don't forget birth and death transitions. In some semesters, the state diagram might only have a couple of states. Label all transitions and actions. Don't forget that the state must be some attribute in your ORD class diagram.

Try to find out the format/range/size that each attribute can be, and the units, accuracy, and precision the users expect to deal with that attribute in. Show legal syntax, using either BNF or syntax diagrams, of any attributes that might (in future) need to be interpreted/parsed by a computer.

What exceptions/errors should the system check for? Generally, transactions take place by asking for an operation and some object info from the user, checking that the records associated with this key exist (referential integrity), and then accepting inputted add/inquire/delete data. As an example of referential integrity, you must of course check for existence of a flight before acting on a transaction which reserves a space on that flight. Specify this must be checked in your brief description of each operation so that the designers of the user manual in the next assignment know to show what happens when the flight doesn't exist. What roughly should the system do in each exceptional case? For instance, you should specify whether there is a need (there usually is) to check, when entering a new entity, if there is an identical one already entered (this ensures continued key uniqueness).

You will likely have to brainstorm iteratively with the group through the various steps above. You may have to iterate your design several times. Bring lots of paper, pencils (not pens), and erasers to meetings. Or find an empty classroom blackboard, and the team leader records the resultant design.

Your specification can use hand drawn figures, but they must be very neat and dark enough to be readable. Do NOT spent a lot of time learning to use some computer drawing tool for this! Draw your original in dark pencil, then hand in a good photocopy. That way, on a subsequent assignment, you can easily change the original with an eraser (keep a photocopy of the original for yourself though as you may need to remember what you handed in on the last assignment before it is marked and returned)! Many students also use the drawing tool in MS-Word, which, surprisingly is hard to find. Try Insert>Object>Microsoft Word Picture.
Also, I don't mind dot-matrix printed documents at all, or even if each chapter comes from a different printer! I'm more concerned with breadth, clarity, and definitiveness of the specifications. e.g. In a flying school, estimate the amount of disk space required for a 10 plane, 200 student system. Make up some throughput estimates so you know whether the 'number of rentals' field in a monthly summary would need to be a 2, 3, 4, 5, or 6 digit (the latter requires a long integer) number on a printed page. i.e. If there is a maximum of 999 rentals in a month, why use 5 spaces on a printout just because it happens to be stored in a 16 bit integer (-32768..32767)?

Note:

- Do NOT assume the reader of your requirements spec will have read this assignment.
- Do NOT assume you will do the design or implementation of the system. Pretend you are the ANALYST during this assignment, on a short-term contract, and you have to communicate (via the req. spec.) your findings to those who will actually design and implement the system, before you leave for a posting overseas. It must be thorough enough that design can begin without your personal presence.
- Each group need only hand in one specification. Everyone in the group will be given the same mark.
- You may find it VERY helpful to have someone re-read this assignment the day before it is due, to remind them of all the things that are necessary (and which might have been forgotten since you read the assignment two weeks earlier when it was first handed out).

MARKING:

You will be marked out of 50 according to the following guide:
3 marks - release title page, and release history/table pages. Follow the suggested formats!
3 marks - requirements specification document title page, version history page, and table of contents.
4 marks - organization and neatness. This does NOT mean laser prints are needed; it means use nice indenting of lists such as this one, and use a ruler or stencil in your drawings. Also, make sure the character size, weight, serif, underlining, and character spacing technology are consistent throughout the document (It is OK if there are tiny differences between team member's wordprocessors/printers, but these differences should be due to technology, not to lack of attention to a team documentation standard). Finally, you MUST put a labeled tab divider after the release history/table page, and after every other document in the binder that you hand in (for Assignment #1, put an unlabelled tabbed divider at the end of the binder). For assignment #1, you can just use a thin duo-tang binder (the TAs will thank you for not using a large 3 ring binder for the first assignment). But for later assignments when several documents in the release have to be handed in, you will need a larger, 3 ring binder.
5 marks - spelling and grammar (read each others work: it's the only way to detect errors. And use a spelling checker!)
4 marks - understandable by, and aimed at the correct audience (in Req. Spec. case it's very broad: users, customers, programmers, technical writers, and management).
4 marks - concentration on WHAT solution is, not HOW to implement it (unless restrictions exist due to present system). Marks will be taken off if you start to talk about internal design of the system.

12 marks – You need a complete description of the required system. Refer frequently to the DFD context diagram (which shows what actors and objects like printers are outside the system), and to the object relationship diagram. You will be judged on your description of the problem and the proposed solution's organization. Also feel free to explain how wonderfully the system will improve the customer's operations. You will to some extent be marked on how broadly you describe the requirements (e.g. type of user interface suggested, response time, throughput, security, maximum file storage required or available at user's site), and wording re minimum requirements (e.g. OS must be Win95 or later).
12 marks - Use of analytical tools/tables/diagrams/dictionary to definitively document the requirements. Remember to label each relationship line and explain the application reasons for the cardinalities on your ORD.
3 marks - description and reasons for different features of 3 releases: prototype, normal, and future features that the design should be tailored to easily add.

Cmpt 275 Team Info Page for Team Letter “ “

Project Description (consult instructor if other than standard):

_____

_____

Computer Language: _____

Programming Language brand or IDE Brand:_____

Programming and target system Hardware Platform/ OS: _____

Word processor/OS: _____

Team Members:

1)

Name: _____

e-mail: _____

phone: _____

good meeting times or timetable info: _____

_____

2)

Name: _____

e-mail: _____

phone: _____

Good meeting times or timetable info: _____

_____

3)

Name: _____

e-mail: _____

phone: _____

good meeting times or timetable info: _____

_____

4)

Name: _____

e-mail: _____

phone: _____

good meeting times or timetable info: _____

_____

5) Add a 5^{th} person only with permission of the instructor.